Amendments to the claims (this list replaces all prior versions):

1. (currently amended) A method comprising

maintaining a database ~~that stores data~~ persistently on a physical storage medium,

partitioning the database into regions based on characteristics of items of data in the database,

receiving ~~accepting~~ tasks that may be in contention to write data to the respective regions, ~~from task sources, at least some of the tasks having competing requirements for use of regions of the database, data included in a given region not being available for simultaneous access for writing by more than one of the tasks having competing requirements and data included in different regions being available for simultaneous access for writing by more than one of the tasks having competing requirements,~~

~~assigning each of the regions to a respective available processor, each of the regions being assignable to any of the processors,~~

reducing the contention among the received tasks by creating ~~defining, for each of the tasks,~~ jobs that are based on each of the tasks, each of ~~which requires~~ the jobs needing to write data in no more than one of the regions, ~~access to a region that is to be accessed by no more than one of the processors, and~~

for each of the regions, queuing only those jobs that need to write data in the region and executing the queued jobs for each region, one job after another and without contention, and

executing ~~distributing the~~ jobs that are queued for different regions simultaneously, in parallel, and without contention ~~for concurrent execution by the assigned processors~~.

2. (currently amended) The method of claim 1 in which the ~~stored data includes data~~ items of data of the database ~~that~~ comprise objects in an object database.

3. (currently amended) The method of claim 1 in which the ~~stored data includes data~~ items of data ~~that~~ are provided as objects to an object-oriented application.

4. (original) The method of claim 1 in which an object relational broker provides persistent storage of objects for an object-oriented application.

5. (currently amended) The method of claim 1 in which the ~~data~~ database comprises ~~is stored in~~ a relational database with object-oriented extensions.

6. (currently amended) The method of claim 1 in which the database comprises files that persistently store the items of data.

7. (currently amended) The method of claim 1, 2 or 3 in which the number of tasks received ~~accepted from task sources~~ is arbitrarily large.

8. (currently amended) The method of claim 1, 2, or 3 in which the tasks are received from task sources, and the number of task sources ~~from which tasks are accepted~~ is arbitrarily large.

9. (currently amended) The method of claim 1, 2, or 3 also including assigning each of the regions to a respective available processor, each of the regions being assignable to any of the processors ~~in which the regions are organized into contention spaces, the number of contention spaces being no less than the number of available processors~~.

10. (currently amended) The method of claim 9 in which the number of the regions is no fewer than the number of available processors ~~each of the jobs requires write access to data in no more than one of the contention spaces~~.

11. (currently amended) The method of claim 9 in which the number of ~~contention spaces~~ the regions is equal to the number of available processors.

12. (currently amended) The method of claim 9 ~~in which~~ also including organizing the ~~organization of~~ regions ~~into contention spaces maximizes~~ to maximize the throughput of the available processors in executing the jobs.

13. (currently amended) The method of claim [[10]] 9 in which the ~~contention spaces~~ regions are assigned dynamically to the available processors to maximize the throughput of the available processors.

14. (currently amended) The method of claim 1, 2, or 3 in which the tasks are ~~accepted~~ received asynchronously.

15. (currently amended) The method of claim 1, 2, or 3 in which the tasks are ~~accepted~~ received concurrently.

16. (currently amended) The method of claim ~~1, 2, or 3~~ 9 in which the processors do not use shared memory.

17. (currently amended) The method of claim 1, 2, or 3 in which ~~defining~~ creating the jobs ~~for each task~~ comprises ~~defining~~ creating a hierarchy of subtasks in which the lowest level of the hierarchy contains the jobs.

18. (original) The method of claim 1, 2, or 3 in which at least one of the tasks comprises a single job.

19. (currently amended) The method of claim 1, 2, or 3 ~~also including a job~~ in which one of the jobs generates ~~generating~~ a task to be performed.

20. (currently amended) The method of claim 1, 2, or 3 in which each of the tasks is completed with a certainty that is at least as high as the certainty that items of data updated in a requested database transaction are ~~is~~ not lost once the transaction is committed.

21. (currently amended) The method of claim 1, 2, or 3 in which at least one of the regions ~~region~~ comprises a single data item.

22. (currently amended) The method of claim 1, 2, or 3 in which at least one of the ~~region~~ regions comprises at least a million data items

23. (canceled).

24. (currently amended) The method of claim 9 in which more than one of the ~~contention spaces~~ regions is associated with one of the available processors.

25. (original) The method of claim 24 in which each of the processors comprises a physical processor running at least one process.

26. (original) The method of claim 1, 2, or 3 in which each of the tasks is generated by a user request.

27. (currently amended) The method of claim 9 in which each of the ~~contention spaces~~ regions is associated with at least two available processors, one of which executes jobs and the other of which performs administrative functions with respect to the associated region ~~contention space~~.

28. (currently amended) The method of claim 1, 2, or 3 in which ~~the distributing of~~ queuing the jobs includes maintaining a queuing system that has a capacity to receive jobs for execution at an arbitrarily large rate by adding processors proportionately to the number of jobs expected to require execution.

29. (original) The method of claim 28 in which the queuing system includes conceptual rows each of which can receive the jobs.

30. (original) The method of claim 29 in which each of the rows is locked when jobs are being received in the row.

31. (original) The method of claim 30 in which a job can be accepted by the corresponding processor for execution from any of the rows that is not locked.

32. (currently amended) The method of claim 9 in which each region comprises millions of sub-regions ~~regions belong to a contention space~~.

33. (original) The method of claim 1, 2, or 3 in which additional jobs are created in connection with the execution of the jobs.

34. (original) The method of claim 33 in which further jobs are created by the additional jobs.

35. (original) The method of claim 33 in which the creation of the additional jobs is dependent on data read from the database in executing the jobs.

36. (currently amended) The method of claim 9 in which additional jobs are created in connection with the execution of the jobs, and a process running on one of the processors executes the jobs and creates the additional jobs, and in which at least some of the additional jobs are queued for regions ~~distributed among contention spaces~~ served by other processors.

37. (original) The method of claim 1, 2, or 3 in which the tasks relate to commercial transactions.

38. (currently amended) The method of claim 9 in which each of the jobs is assigned an index associated with the corresponding region ~~contention space~~.

39. (currently amended) The method of claim 38 in which the indexes are used to load balance the jobs among the processors.

40. (original) The method of claim 1, 2, or 3 in which the database includes database units that are distributed among different physical locations.

41. (original) The method of claim 1, 2, or 3 in which each of the jobs comprises steps.

42. (original) The method of claim 41 in which execution of a job includes executing a portion of the steps, committing a database transaction representing those steps, and repeating until the job is completed.

43. (original) The method of claim 42 also including, upon a failure to complete any portion of the steps, restarting the execution at the first step of the failed portion with at least the same level of certainty that the job will be completed as the certainty that data written in a requested transaction is not lost once the transaction is committed.

44. (original) The method of claim 31 in which a row is locked only for reading when a processor is accepting jobs from that row.

45. (currently amended) The method of claim 9 in which (1) queuing ~~the distributing of~~ the jobs includes maintaining a queuing system that has a capacity to receive jobs for execution at any arbitrarily large rate, (2) the queuing system includes conceptual rows each of which can receive the jobs, and (3) the queuing system includes conceptual columns associated with respective contention spaces, the queuing system comprising a conceptual matrix of cells at the intersections of the rows and columns, in which each of the cells may be read from or written to without conflicting with reads and writes to other cells.

46. (original) The method of claim 45 in which the rows are associated with sources of jobs and the number of rows is sufficient to permit all of the sources of jobs to load jobs into the queue concurrently.

47. (original) The method of claim 45 in which the number of rows is sufficient to permit jobs to be fetched for execution from all of the columns concurrently.

48. (original) The method of claim 1, 2, or 3 also including synchronizing the executions of synchronization groups of the jobs to ensure correctness of results.

49. (original) The method of claim 48 in which the synchronizing includes assigning to each of the jobs of a synchronization group a tag that identifies them as members of the group.

50. (original) The method of claim 48 in which the synchronizing includes assigning to each of the jobs of a synchronization group a quorum fraction representing the job's proportion of participation in the group.

51. (original) The method of claim 50 in which the jobs are not executed until all of the jobs in the synchronization group are ready for execution by a processor.

52. (currently amended) Apparatus comprising

a physical storage medium on which a database ~~that stores data~~ is persistently maintained, the database being partitioned into regions, and

a job processing mechanism that (1) ~~accepts~~ receives ~~an arbitrarily large number of tasks asynchronously from an arbitrarily large number of task sources, at least some of the~~ tasks that may be in contention to write data to the respective ~~having competing requirements for use of~~ regions of the database, ~~data included in a given region not being available for simultaneous access for writing by more than one of the tasks having competing requirements and data included in different regions being available for simultaneous access for writing by more than one of the tasks having competing requirements,~~ (2) creates jobs that are based on each of the tasks, each of the jobs needing to write data in no more than one of the regions ~~organizes the regions into non-conflicting contention spaces each associated with a different available processor~~, (3) for each of the regions, queues only those jobs that need to write data in the region, one job after another and without contention, ~~decomposes each of the tasks into jobs each of which requires write access to regions belonging to no more than one of the contention spaces,~~ and (4) executes jobs that are queued for different regions simultaneously, in parallel, and without contention ~~distributes the jobs to the corresponding contention spaces for concurrent execution by the associated processors~~.

53. (currently amended) The apparatus of claim 52 in which the ~~stored data includes~~ data items of the database ~~that~~ comprise objects in an object database.

54. (currently amended) The apparatus of claim 52 in which the ~~stored data includes~~ data items ~~that~~ are provided as objects to an object-oriented application.

55. (original) The method of claim 52 in which an object relational broker provides persistent storage of objects for an object-oriented application.

56. (original) The method of claim 52 in which the data is stored in a relational database with object-oriented extensions.

57. (currently amended) A physical article or object bearing instructions to cause a processor to

execute a job requiring access to a region of a database that is ~~stores data~~ persistently maintained on a physical storage medium, the job including instructions and pointers to data in the region of the database,

the job that is executed being selected from a ~~contention space~~ queue of jobs identified by an index, the jobs in the ~~contention space~~ queue being in contention to write data ~~having~~

~~competing requirements~~ to ~~write into~~ the region of the database, the index distinguishing the queue of jobs ~~contention space~~ from other ~~contention spaces~~ queues of jobs that do not have need ~~competing requirements~~ to write data into the region of the database.

58. (currently amended) The physical article or object of claim 57 in which the ~~stored data includes data items of the~~ database comprises items of data that comprise objects in an object database.

59. (currently amended) The physical article or object of claim 57 in which the ~~stored data~~ database includes data items that are provided as objects to an object-oriented application.

60. (previously presented) The physical article or object of claim 57 in which an object relational broker provides persistent storage of objects for an object-oriented application.

61. (currently amended) The physical article or object of claim 57 in which the database comprises ~~data is stored in~~ a relational database with object-oriented extensions.

Claims 62 – 67 (cancelled)

68. (currently amended) A method comprising

maintaining a database ~~that stores data~~ persistently stored on a physical storage medium, the database providing ~~and provides~~ a primary level of guarantee that data items written in a requested transaction ~~is~~ are not lost once the transaction is committed,

partitioning the database into regions based on characteristics of the items of data,

receiving ~~accepting~~ tasks from task sources for concurrent execution by multiple processors, at least some of the tasks being in contention ~~having conflicting requirements~~ to write data in ~~into~~ one region ~~the same region~~ of the database, and

providing a software mechanism that guarantees, ~~as~~ at least to the primary level of guarantee, that the tasks will be executed ~~and will be executed~~ without loss of any of the data items, ~~and~~ by translating each task into jobs, each job needing to write data in no more than one of the regions ~~without the occurrence of any actual conflict with respect to the region~~ of the database.

69. (currently amended) The method of claim 68 in which the ~~stored data includes~~ data items ~~of the database that~~ comprise objects in an object database.

70. (currently amended) The method of claim 68 in which the ~~stored data includes~~ the data items ~~that~~ are provided as objects to an object-oriented application.

71. (original) The method of claim 70 in which an object relational broker provides persistent storage of objects for an object-oriented application.

72. (currently amended) The method of claim 70 in which the data items are ~~is~~ stored in a relational database with object-oriented extensions.

73. (original) The method of claim 68, 69, or 70 also including sending to the task source an acknowledgement of acceptance of the task.

74. (original) The method of claim 68, 69, or 70 also including sending to the task source a notification after completion of the accepted task.

75. (currently amended) The method of claim 68 in which the ~~task is decomposed into~~ jobs ~~that~~ are executed by different ones of the multiple processors in a manner that prevents any actual conflict between jobs.

76. (currently amended) The method of claim [[74]] 68 in which the jobs are subjected to a synchronization mechanism that enables a determination of the completion of a task.

77. (original) The method of claim 76 in which the synchronization mechanism includes a tag that identifies a job as participating in a group of jobs.

78. (original) The method of claim 76 in which the synchronization mechanism includes a quorum fraction that represents the job's proportion of participation in the group.

79. (original) The method of claim 77 also including determining whether the quorum fractions of all of the jobs of a group add to a completed quorum.

80. (original) The method of claim 75 in which the task is notified of completion when all of the jobs have been completed.

81. (original) The method of claim 75 in which the task is assigned to a contention space.

82. (original) The method of claim 75 in which completion notification jobs are assigned for execution in the same contention space as the task.

83. (previously presented) The method of claim 68 in which the database comprises an object-oriented database.